RL-TR-94-96
Final Technical Report
July 1994

O-PLAN2 VS SIPE-2 - A GENERAL COMPARISON

Northeast Consortium for Engineering Education

Dr. Carla Ludlow

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

Rome Laboratory
Air Force Materiel Command
Griffiss Air Force Base, New York

This report has been reviewed by the Rome Laboratory Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RL-TR-94-96 has been reviewed and is approved for publication.

APPROVED: Karen M. Alguire

KAREN M. ALGUIRE
Project Engineer

FOR THE COMMANDER: John A. Graniero

JOHN A. GRANIERO
Chief Scientist
Command, Control & Communications Directorate

If your address has changed or if you wish to be removed from the Rome Laboratory mailing list, or if the addressee is no longer employed by your organization, please notify RL (C3CA) Griffiss AFB NY 13441. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	July 1994	Final Feb 93 - Feb 94

4. TITLE AND SUBTITLE
O-PLAN2 VS SIPE-2 - A GENERAL COMPARISON

5. FUNDING NUMBERS
C - F30602-93-C-0017
PE - 62702F
PR - 5581
TA - 27
WU - PK

6. AUTHOR(S)
Dr. Carla Ludlow

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Northeast Consortium for Engineering Ducation
68 Port Royal Sq
Port Royal VA 22535

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Rome Laboratory (C3CA)
525 Brooks Road
Griffiss AFB NY 13441-4505

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
RL-TR-94-96

11. SUPPLEMENTARY NOTES
Rome Laboratory Project Engineer: Karen M. Alguire/C3CA/(315) 330-4833

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

This report addresses strengths and limitations detected in SIPE-2 and O-Plan2 as a result of some experiments that were performed at Rome Laboratory using those generative planners. The main objective of the project was to provide an understanding of generative planners and familiarization with O-Plan2 and SIPE-2. A comparison between O-Plan2 and SIPE-2 was a secondary objective, mainly a subproduct of the experimentation with both planners.

14. SUBJECT TERMS
Generative Planners, Artificial Intelligence, Nonlinear Planning Systems, Plan Specification, Plan Execution, Plan Refinement

15. NUMBER OF PAGES
40

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Contents

1 Introduction

This paper reports some experiments that were performed at Rome Laboratory using two generative planners: O-Plan2 ([O-Plan2 93], [Currie & Tate 91]) and SIPE-2 ([Wilkins 93], [Wilkins 88]). O-Plan2 was developed by AIAI-University of Edinburgh. SIPE-2 was developed by SRI International, California. The team that participated in the project consisted of: Carla O.Ludlow, as consultant to Rome Laboratory, Karen Alguire and Albert Franz, from Rome Laboratory. Albert Franz was involved in the project for only two months. The main objective of the project was to provide the in-house team with an understanding of generative planners and familiarization with O-Plan2 and SIPE-2. A comparison between O-Plan2 and SIPE-2 was a secondary objective, mainly a subproduct of the experimentation with both planners.

In order to become familiar with the systems, we started by going through the demonstration examples that come with O-Plan2 and SIPE-2. The next step was to solve problems not encoded in either of the systems, starting with *simple* classical AI planning problems. At that time, it seemed obvious to us that the Missionaries and Cannibals puzzle (MC puzzle) would qualify as a *simple* classical AI planning problem. We encountered some problems in solving the MC puzzle in SIPE-2 and we could not solve it with the current version of O-Plan2, as it does not handle numerical calculations [1]. However, the Missionaries and Cannibals puzzle constituted an excellent vehicle for testing and evaluating several features of the planners. In particular, the MC puzzle was very inspiring regarding the generation of several small examples that were used to test two aspects of the planners: deductive reasoning and goal phantomization. The paper "Looking at O-Plan2 and SIPE-2 Through the Missionaries and Cannibals" ([Ludlow & Alguire 94]) has a detailed description of the issues related to the implementation of the MC puzzle in SIPE-2 and O-Plan2. Alguire [Alguire 94] gives a detailed description of the encoding experience with SIPE-2 as a result of this project.

This report addresses strengths and limitations detected in SIPE-2 and O-Plan2 as a result of our experiments. This paper only addresses plan refinement. Temporal reasoning is not considered. Furthermore, since the current version of O-Plan2 only includes a very simple model for resource reasoning (only consumable resources), resource analysis was also excluded.

Section 2 describes installation issues as well as it gives the history of the different versions of O-Plan2 and SIPE-2 installed on Rome Laboratory's machines. Section 3 gives an overview of O-Plan2 and SIPE-2. Section 4 describes representational issues regarding objects, predicates

[1] When we selected the MC puzzle we were convinced that the O-Plan2 could handle numerical calculations. The documentation of O-Plan2 includes the "full" Task Formalism (TF) syntax that the O-Plan2 team is targeting to support in a final O-Plan2 version in order to demonstrate how their designs and ideas are developing. At times we found it difficult to distinguish what is implemented in the current version from what is planned to be integrated in future versions of O-Plan2. Since one of our objectives was to evaluate the way the planners handle numerical calculations, we excluded the idea of implementing arithmetics in O-Plan2 ourselves using symbolic O-Plan2 functions.

and operators. Section 5 addresses implementation of context dependent effects, section 6 phantomization of goals, section 7 numerical reasoning, and section 8 search. In section 9 I present a synopsis of the conclusions of the experimentation performed with SIPE-2 and O-Plan2 at Rome Laboratory.

2 Installation Issues & History

O-Plan2

We started the project using version 1.1 of O-Plan2, the first version of O-Plan2 that was provided to Rome Laboratory, implemented in Kyoto Common Lisp. When we started experimenting O-Plan2, version 1.1, we soon realized that version had some limitations, in particular regarding the description of the format of the file that contains the plan generated by the planner. The documentation of O-Plan2 did not contain a complete description of the output file. We contacted AIAI-University of Edinburgh, the authors of O-Plan2, and we were told that they were about to release a new version of O-Plan2, in which the output limitations would be overcome. We got a new release of O-Plan2 at the beginning of June. We had several problems to install that version due to incompatibilities with the Sun operating system on the Rome Laboratory's Sun workstations, version 4.4. The Sun operating system of Rome Laboratory's Sun workstations was upgraded in August, 1993. Concomitantly, a new version of O-Plan2 was out. It was version 2.1. We installed it successfully. All the experimentation we performed with O-Plan2 was done with version 2.1.

SIPE-2

We started the project using SIPE-2 version 4.3. As part of the experimentation performed by the Rome Laboratory team several errors were detected in SIPE-2. All the errors have been fixed with "patches" or a new release. Below is a list of the major errors/patches that were detected and fixed in SIPE-2.

- checking the argument order of an operator
- "not" goals
- numerical capabilities - matching numbers with numerical instantiated variables
- backtracking when using interactive planning
- "or" predicates in precondition
- proper loading of patches
- popup menus
- printing of operators

Version 4.3 of SIPE-2 was upgraded to version 4.4, our current version.

3 Overview of O-Plan2 and SIPE-2

O-Plan2

O-Plan2 is an environment for plan specification (Task Assignment), planning, and plan execution support. O-Plan2 is an ongoing project. The current version 2.1 concentrates on plan refinement using a domain-independent, hierarchical, and nonlinear approach with a simple plan dispatching execution system. O-Plan2 uses planning variables with constraint posting. The current system integrates temporal constraints and a simple consumable resources model. Time specifications act as constraints on the search conducted by the planner, with time window propagation being conducted by the time point manager. O-Plan2 has been designed for distributed implementation on geographically distributed systems (at the agent level), on coarse grained multi-processors (at the component level of a single agent - e.g. the planner) and for fine grained massively parallel processors or special purpose hardware (at the constraint manager level).

Version 2.2 of O-Plan2 with resource oriented demonstration is planned to be delivered in July 94 and version 2.3, which integrates planning, execution and re-planning, is planned to be delivered in July 95.

O-Plan2 [Tate *et al* 94b] has an agenda driven blackboard architecture in which each control cycle can post pending tasks during plan generation. Different knowledge sources with different skills are responsible for processing the pending tasks from the agenda. O-Plan2 manipulates a plan state which is a data structure containing the emerging plan, the flaws remaining in it and the information used in building the plan. The main components are:

1. Domain Information - the information which describes an application domain and tasks in that domain to the planner in a structured language.
2. Plan State - the emerging plan to carry out identified tasks.
3. Knowledge Sources - the processing capabilities of the planner (*plan modification operators*).
4. Support Modules - functions and constraint managers which support the processing capabilities of the planner and its components.
5. Controller - the decision maker on the *order* in which processing is done.

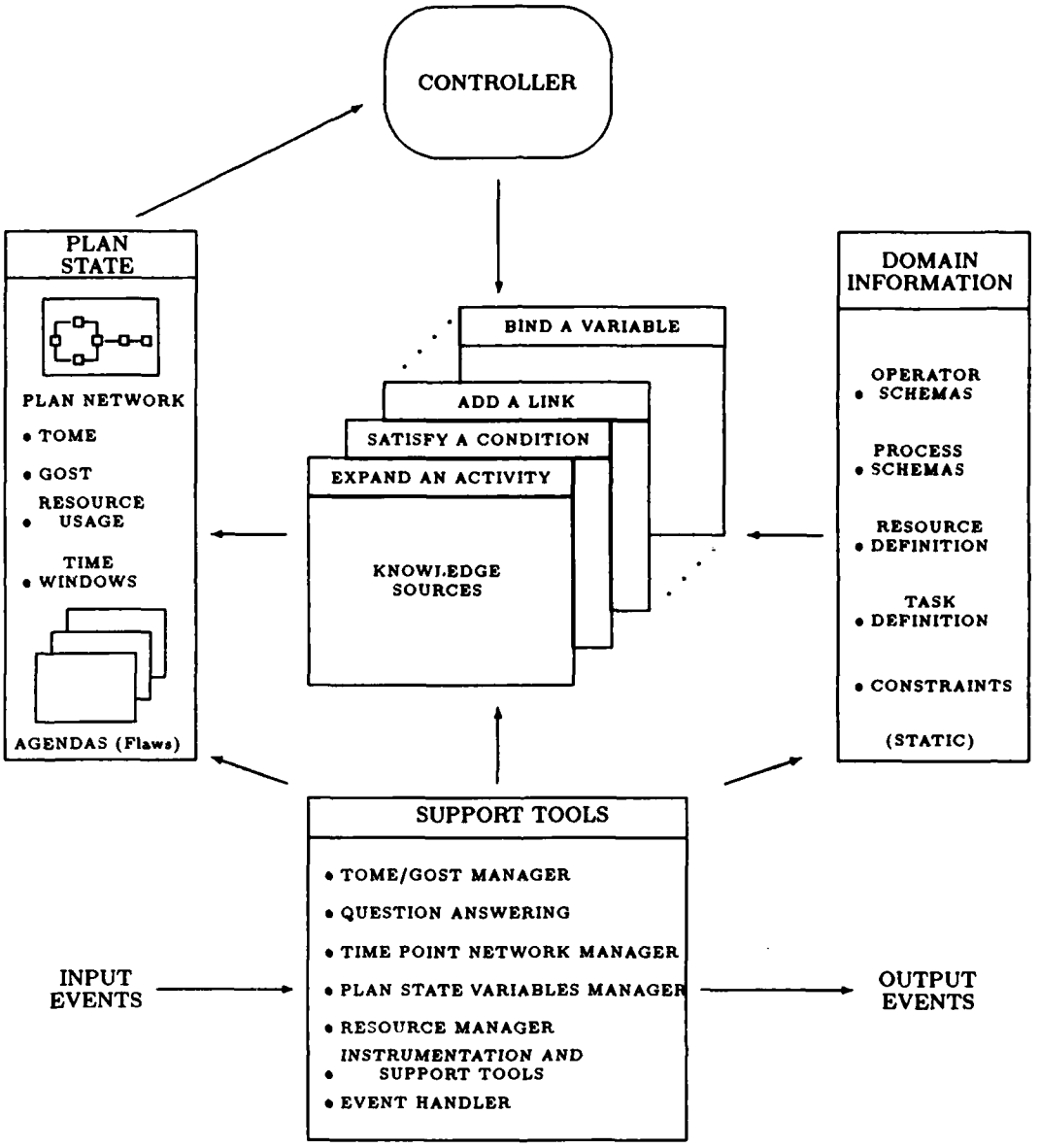
The plan state is the dynamic data structure corresponding to the plan in progress and it consists of the plan network, the plan causal structure (sometimes called the *teleology* of the plan), and the agenda list. The plan network is a partially ordered network of activities. It is the basis of the plan representation. The plan information is concentrated in the "Associated Data Structure" (ADS). The ADS is a list of node and link structures noting temporal and resource information, plan information and a plan history. The plan causal

structure keeps explicit information about the rationale of the plan. It includes the Goal Structure (GOST) and the Table of Multiple Effects (TOME) which provide support to the condition achievement support module (Question Answerer or QA) used in O-Plan2. The agenda list(s) contains the necessary information for the resolution of pending "flaws" of the current plan that prevent it from being capable of execution. "Flaws" may also represent potentially beneficial, but as yet unprocessed, information.

The Knowledge Sources are the processing units associated with the processing of the flaws contained in the plan and they embody the planning knowledge of the system. There are as many knowledge sources (KSs) as there are flaw types, including the interface to the user wishing to exert an influence on the plan generation process.

In O-Plan2 there are a number of support modules that are intended to provide efficient support to a higher level where decisions are taken. They are intended to provide complete information about the constraints they are managing or to respond to questions being asked of them to the decision making level itself. The support modules include the Time Point Network (TPN) Manager to manage metric and relative time constraints in a plan, the Question-Answerer (QA), TOME and GOST Management (TGM) to manage the causal structure (conditions and effects which satisfy them) in a plan, the Plan State Variables Manager to manage partially bound objects in the plan, the Resource Utilization Management to monitor and manage the use of resources in a plan, and Instrumentation and Diagnostics routines. The QA is a central piece of O-Plan2's condition achievement procedure. It answers the basic question of whether a proposition is true or not at a particular point in the plan. The answer it returns may be (i) a categorical "yes", (ii) a categorical "no", or (iii) a "maybe", in which case QA will supply an alternative set (structured as a tree) of strategies which a knowledge source can choose from in order to ensure the truth of the proposition.

O-Plan2 Architecture



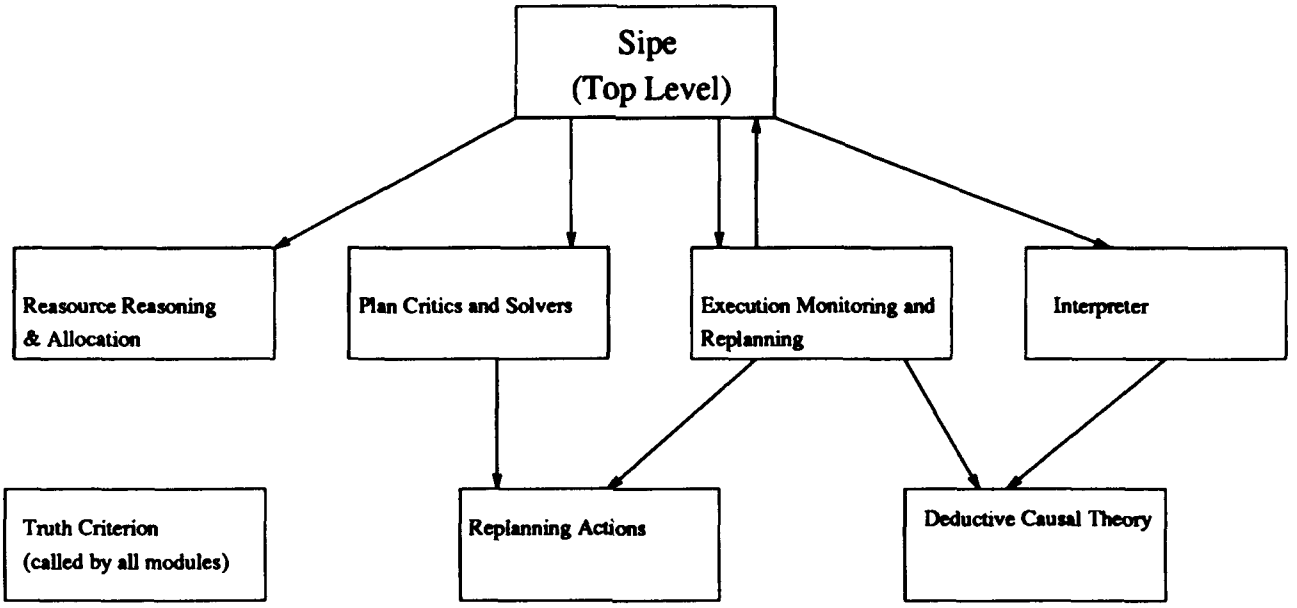
SIPE-2

SIPE-2 is a domain-independent, hierarchical, and nonlinear AI generative planning system with mechanisms for reasoning about context-dependent-effects. It uses planning variables and the ability to post constraints on them. SIPE-2 has some replanning capabilities and also some capability to reason for reasoning about resources. Temporal reasoning in SIPE-2 is very limited. Time is treated as a consumable resource that can be consumed but not produced, and whose consumption over parallel tasks is nonadditive. Each action specification may have a start-time, stop-time, and duration slots containing variables with numerical constraints on them that are satisfied by the planner. SIPE-2 has been extended to interface with General Electric's Tachyon system for temporal reasoning. Temporal constraints in SIPE-2 are written to a file that is read by Tachyon. Tachyon processes these constraints and writes a file of narrowed time windows for the start-times, end-times, and durations of all the actions. This information is used to update the SIPE-2 plan before planning continues.

SIPE-2 is a component of CYPRESS, a system that supports planning capabilities including action specification, generative planning, reasoning about uncertainty, reactive plan execution, and dynamic replanning.

SIPE-2's architecture is not as modular as O-Plan2's architecture. The following figure depicts a conceptual division of SIPE-2 into different modules. However, as Wilkins stresses ([Wilkins 88]), this division is primarily for expository purposes since in the actual code there is not such a sharp demarcation separating modules.

SIPE-2 Architecture



The planner is controlled by the top level mechanism. In particular, the top-level algorithm determines when to check for resource conflicts, when to apply plan critics, and when to call the interpreter to elaborate the plan by applying an operator to one of the goals in the plan.

The plan critics are responsible for finding problems in the plans produced, checking whether the global constraint network is satisfiable, finding resource conflicts, checking which goals are already true (phantoms), finding problematic interactions between unordered actions, and for correcting the problems encountered. The critics responsible for resource reasoning are individualized from other plan critics due to their importance and novelty within the context of generative planners.

The execution module accepts descriptions of arbitrary unexpected occurrences and it then determines how these occurrences affect the plan being executed, possibly modifying the plan by removing certain subplans and inserting certain goals. When the revised plan contains unsolved goals the execution module calls the search algorithm to expand the plan. The execution module invokes the causal theory to deduce effects of the unexpected occurrences. Both the execution module and the plan critics use the replanning actions to alter existing plans.

The deductive causal theory module is responsible for deducing context-dependent effects. In section 5 I highlight this feature of SIPE-2.

The Truth Criterion determines whether a formula is true at a particular point in time. All modules of the system depend on the Truth Criterion and Wilkins claims that its heuristics help to avoid exponential work on an NP-complete problem ([Wilkins 88]. In section 6 I describe a set of situations where SIPE-2's heuristics lead to false judgements by the Truth Criterion. It would be interesting to do a exhaustive study to determine the frequency of false judgment by the Truth Criterion as a result of its heuristic implementation.

The interpreter interprets the plan language in which the operators are written.

4 Representational Issues

In SIPE-2 [Wilkins 93], objects are represented combining a frame-based representation with predicate calculus. The *sort hierarchy* describes the invariant properties of perpetual objects and their structure in terms of classes with inheritance properties. Properties of the objects that can vary are represented using predicate calculus. In O-Plan2 ([O-Plan2 93], [Drabble 93]) objects are defined as *types*. Since O-Plan2 is not provided with a hierarchical structure of objects, defining hierarchical relationships and inheritance properties between objects is not as easy as in SIPE-2. In SIPE-2 predicates are described in first-order logic. Static and dynamic predicates are treated differently. O-Plan2 offers more flexibility in this area, representing predicates via *patterns*, whose value is expressed functionality, e.g., {*on a b*} = *true* or {*weather Rome*}= *snow*. Always statements define static predicates, e.g., *always* {*capital Portugal*}= *Lisbon*.

Operators in SIPE-2 correspond to the actions that can be taken in the world. O-Plan2 refers to them as *operator schemas* or simply *schemas*. SIPE-2 includes special types of operators called *Causal rules* and *state rules* which describe causal connections of the domain. In section 5 I discuss deductive causal theories. Variable declarations are performed through the *arguments* in SIPE-2 and *vars* field in O-Plan2. Constraints on the values of variables can be declared in the argument slot in SIPE-2, while O-Plan2 uses a different field, *vars-relations*. Operators and schemas have two forms of being triggered. In SIPE-2 the *purpose* of an operator determines which goals the operator can solve. The condition type *only-use-for-effects* in O-Plan2 is equivalent to the slot purpose in SIPE-2. Operators and schemas can also be triggered to expand some action to a lower level of detail. In SIPE-2, an operator can be expanded to a lower level with other operators that appear in its plot as *process* slots or as *choiceprocess* (see description of plot below). In other words, if an operator A has as process the operator B, operator A will be expanded with operator B. Choiceprocess is a way in SIPE-2 to specify a sequence of alternative operators to expand a certain operator. In the case of O-Plan2 the slot *expands* is equivalent to the slot process in SIPE-2. We did not encounter in O-Plan2 an equivalent to choiceprocess. In either of the system, when the purpose or only-use-for-effects are the trigger, the selection of operators are based on a single effect at a time, i.e, even if operators have multiple effects, they are triggered based on a single effect. Note, however, if several operators or schemas match an effect, then each is chosen and made an alternative in the search for a solution. In either of the systems, if there are more than one operator to expand a node satisfying all the conditions, the criterion of selection is the order of appearance in the file, except in the case where choiceprocesses are defined in SIPE-2.

In SIPE-2, *preconditions* of an operator are predicates that must be true. *Preconditions* in SIPE-2 can also be used for determining the current values of variables. O-Plan2 [Tate 93] uses different condition types for schema selection. In September 93 when we were trying to fully understand the differences between each condition type we were told by the O-Plan2 team that condition types are still an area of research (and even change) in O-Plan2. Our

understanding is that *only-use-if* and *only-use-for-query* in O-Plan2 are similar to the slot *preconditions* in SIPE-2. *Only-use-if* is a filter that defines the conditions that must be true in order for the schema to be applied. *Only-use-for-query* is a mechanism to establish current values of variables. O-Plan2 also includes the condition type *supervised*, a condition used to protect effects which are established within the scope of the schema, and *unsupervised*, a condition that specifies a constraint on the schema which is normally satisfied externally, i.e., it does not normally use the resources of the schema. The equivalent conditions in SIPE-2 are *protect-until* and *external*. Both systems allow the definition of constraints in terms of resources and temporal constraints [2]. SIPE-2 allows the specification of conditions through the definition of lisp functions. In O-Plan2, *Compute-conditions* are anticipated to be provided in a future release. *Goals* in SIPE-2 correspond to the conditions to be satisfied by the planner. The equivalent concept in O-Plan2 is *achieve* conditions. O-Plan2 allows the additional definition of *achieve at N* which adds flexibility in terms of encoding. In section 6 I describe a set of situations where O-Plan2 and SIPE-2 behave differently when a goal (achieve condition) is posted.

The set of statements that define the form of expansion of an operator is referred to the *plot* in SIPE-2. In O-Plan2, these are referred to as *nodes* and their associated *orderings*. In both systems, the form of expansion of an operator is defined by a network of nodes with associated orderings, conditions and effects (e.g., passing of variables, resource constraints, time constraints, effects, supervised or protect-until, unsupervised or external conditions, achieve, etc). SIPE-2 provides a loop facility for parallel actions. The definition of a task or problem is done by means of a *task schema* in O-Plan2 and a *problem* definition in SIPE-2. The task schema defines a set of nodes (actions) and corresponding orderings, conditions, constraints and effects at each node and effects. The predicates that define the initial state of the world are part of the task definition in O-Plan2. This feature is handy for debugging purposes, when one wants to consider different initial states. In SIPE-2, a problem definition corresponds basically to the plot of an operator as a set of goals, with the ability to include variables and constraints.

The next sections highlight the comparison of O-Plan2 and SIPE-2 considering the following aspects: deductive causal theories, goal phantomization, numerical reasoning, and search.

[2] We did not test the usage of resource constraints nor temporal constraints.

5 Deductive Reasoning

One of the key features of SIPE-2 is the possibility of encoding a causal theory to represent and reason about the effects of actions in different world states. SIPE-2 allows the separation of knowledge about actions from knowledge about causality and thus, SIPE-2 provides a mechanism to implement a causal theory of the particular domain ([Wilkins 88], [Wilkins 93]). In SIPE-2 the actions that one might take are represented by means of operators and domain rules. The effects of actions are expressed through the effect slot of operators except context-dependent effects which are encoded using the deductive rules.

SIPE-2 deductive rules have triggers, preconditions, conditions, and effects. A deductive rule is applied when a given effect matches the trigger of a rule. If the precondition and condition of the rule hold, the effects of the rule are added to the current state of the world. Triggers and conditions of a rule are always matched in the current state of the world while preconditions are matched in the previous state. SIPE-2 deductive rules are divided into three types: init-operators are applied only once to deduce new statements about the initial world model; causal rules and state rules are rules that deduce new statements about the world. The only difference between causal rules and state rules is the order of applicability - causal rules are applied before the state rules. Causal rules and state rules can be triggered recursively.

As an example of a causal rule in the MC puzzle, the following rule expresses that if the number of cannibals increases on one bank by some amount, the same amount of cannibals decreases on the opposite bank.

```
causal-rule: produce-and-consume-cannibals
arguments: cannibals, river-bank1, numerical3, river-bank2;
trigger: (produce cannibals river-bank1 numerical3);
precondition: (opposite-bank river-bank1 river-bank2)
effects: (consume cannibals  river-bank2 numerical3);
end causal-rule
```

In the current version of O-Plan2 there is no implemented mechanism to represent a theory of causality, in particular there is no mechanism equivalent to the domain rules in SIPE-2. As a result, all the effects of an action have to be explicitly stated. Furthermore, actions that have different effects depending on the particular state of the world have to be encoded through a multiplicity of operators, each operator corresponding to a possible situation in which the action might take place. Let us consider a simple example of a domain theory encoded in SIPE-2.

```
causal-rule: one-location-at-a-time-object
arguments: object1, location1, location2 is not location1;
trigger: (at object1 location1);
```

```
precondition: (at object1 location2);
effects: (not (at object1 location2));
end causal-rule

causal-rule: move-contents
arguments: object1, location1, object2 class universal;
trigger: (at object1 location1);
precondition:  (in object2 object1);
effects: (at object2 location1);
end causal-rule

causal-rule: move-object-held
arguments: person1, location1, object1, location2 is not location1;
trigger: (at person1 location1);
precondition: (holding person1 object1);
effects: (at object1 location1);
end causal-rule
```

In this simple domain theory, three rules are considered. The first rule states that objects can only be at one location at a time. The second rule states that if an object is moved from location1 to location2 so are its contents. The third rule states that if a person moves from one location to another so do all the objects that (s)he is holding. As an example of a set of simple objects that would give an interpretation to this theory we can think of a briefcase, an object that can contain several objects; a pencil case; an object that can be in a briefcase; keys; etc. Let us add to our theory the following operators:

```
operator: pick-up-object
arguments: person1, object1, location1;
purpose: (pick-up person1 object1);
precondition: (at person1 location1),
              (at object1 location1);
plot:
    process
action: pick-up
        arguments: person1, object1;
        effects: (pick-up person1 object1),
                 (holding person1 object1);
end plot end operator

operator: commute-from-location-to-location
arguments: person1, location1, location2 is not location1;
purpose: (at person1 location1);
```

```
precondition: (at person1 location2);
plot:
    process
        action: commute
        arguments: person1, location1;
        effects: (at person1 location1);
end plot end operator
```

How would one translate the operator Commute-from-location-to-location into O-Plan2 TF language? Several operators would have to be considered in order to keep track of all the different effects corresponding to the different situations. Each different situation requires a different operator: person is not holding anything; person is holding one object; person is holding n objects; person is holding one object that contains one object; etc. One can see that the number of operators required can grow very quickly which has a high cost in terms of the time required to encode a problem and also in terms of search. In general if an action has n causal effects, if the strict STRIPS assumption is adopted, up to 2^n operators might be required to encode that action. If domain rules can represent the n effects a single operator and n rules can represent the action. Moreover, the same rules will probably be used for inference of other causal effects of other actions.

Within the SOCAP (System for Operations Crisis Action Planning) [Wilkins & Desimone 92], several domain rules are implemented regarding commutativity, reasoning about location and different level of abstraction of locations, and reasoning about movement of aggregate forces.

Wilkins [Wilkins 88] addresses two potential problems with domain rules. The first problem has to do with conflicting deductions. The default implemented in SIPE-2 is to keep the first deduction and ignore contradictory deduction, though in some situations one may want to change that default. An example of that occurs in the mobile-robot domain. The second problem relates to the need to instantiate some variables in order to match the precondition of a domain rule. The default in SIPE-2 is to constrain variables in order to match a domain rule, but only when the two variables are already constrained to be of the same class. If a domain rule requires further specification of a variable's class, it will fail.

A limitation of the deductive rules in SIPE-2 is the fact that they consider only a single trigger. Situations occur when causality requires two or more effects to happen simultaneously.

6 Goal Phantomization

As part of our experiments related to the Missionaries and Cannibals puzzle, we performed some tests comparing the way O-Plan2 and SIPE-2 phantomize goals, i.e., the process of *achieving* a goal by having it already true at the point in the plan where it occurs. The term phantomization came about with Nonlin and NOAH. O-Plan2 does not put any goal or condition into its plan network, so it does not need phantoms in the network, rather it only introduces actions into the network when they are needed. Otherwise the process is just to find contributors that satisfy conditions. Nevertheless, we still use the term phantomization. A particular situation was detected when we were trying to implement the operators that move two people from one bank to the other. The following problem illustrates an identical situation through a smaller example.

```
class: persons
instances:   nelson,  carla, ken, karen, bryan;
end class

class: places
instances:  rome, ny;
end class

predicates:
(nice nelson), (nice carla), (nice bryan), (nice karen)
(on nelson ny), (on ken ny), (on carla ny)
(wants-ride carla), (wants-ride ken)
end predicates

operator: transport-person-ride;
arguments: person1,  place1, person2 is not person1, place2 is not place1;
purpose: (on person1 place1) ;
precondition: (nice person1),  (on person1 place2),
              (on person2 place2), (wants-ride person2);
plot:

    process
        action: transport-person-prim;
        arguments: person1, person2,  place2, place1;
        effects: (on person1 place1), (not (on person1 place2)),
                 (got-ride person2), (not  (wants-ride person2)),
                 (on person1 place2);
end plot end operator
```

```
problem: prob-ride-parallel
parallel

branch 1: goal: (on nelson rome)
branch 2: goal: (got-ride carla)

end parallel
end problem
```

In this example the first goal, (on Nelson Rome), triggers the operator Transport-person-ride. The primary effect of this operator in relation to the goal (on Nelson Rome) is (on person1 place1) [3]. On the other hand, (got-ride person2) is a secondary effect of the operator Transport-person-ride, in relation to the goal (on Nelson Rome). When SIPE-2 encounters the second goal (got-ride Carla), since the effect (got-ride person2) was achieved as a secondary effect and at the same level as the parallel goal (on Nelson Rome), SIPE-2 does not phantomize it.

Several examples were tested in SIPE-2, regarding the phantomization of goals. The situation tested was the following. A single operator with two main effects - the first main effect matching the purpose of the operator while the second main effect does not match the purpose of the operator. Furthermore, the second main effect is either a predicate of arity zero, i.e., a constant, or a predicate of arity one, i.e., a predicate involving one variable. If the second main effect involves a variable, that variable is not involved in the purpose of the operator. All the problems tested consisted of two goals. The first goal matches the purpose of the single operator (so it is guaranteed to be satisfied). The second goal matches the secondary effect of the main operator. However, since the variables involved in the secondary effect of the operator are different from the variables involved in the purpose of the operator, there is no guarantee that SIPE-2 will instantiate them adequately in order to phantomize the second goal. What I wanted to test with the experiment was under what conditions the second goal would be (or not be) phantomized as a result (secondary effect) of the application of the single operator to achieve the first goal. Two types of problems were considered:

- Type 1 - the goals are sequential
- Type 2 - the goals are in parallel

The table presented below shows a summary of the analysis of goal phantomization in SIPE-2 for different situations (i.e., different values of different parameters) and considering the

[3]The primary effect of an operator is the purpose of the operator, when it is a single purposed operator. In general, the primary effect of an operator is the purpose of the operator that matches the goal that triggered its application. Secondary effects of an operator in relation to some goal are all the effects that are not primary effects.

two types of problems described above. The first column of the table has the number of instances that satisfy the preconditions for the variable of the secondary effects of the main operator, i.e., the effects that involve uninstantiated variables. When the value displayed for the column is "-" it means that the secondary effect (corresponding to the second goal to be satisfied) did not involve any variable. The second column of the table displays the order of appearance in the file of the instance involved in the second goal, in relation to all the instances that satisfied the corresponding secondary effect. As an example, if the value is 2nd it means that, from all the instances that satisfied the secondary effect, the one included in second goal appears in 2nd place in the file where the problem is defined. The third column indicates if the instantiate slot was used in the main operator that achieves the goal, while the fourth column indicates if the operator copy was used[4]. The fifth column indicates in which condition SIPE-2 solves the problems - "sequential" means that SIPE-2 could solve only the sequential problem; "parallel" means that SIPE-2 could solve only the parallel problem; "both" means that SIPE-2 could solve both problems; "neither" means that SIPE-2 could not solve either of the problems. The following example illustrates a case whereby two instances satisfy the preconditions for the variable involved in the secondary effect of the single operator that corresponds to the second goal (carla and ken satisfy the precondition (on person2 place2)). In this example the copy operator is not used but the instantiation slot is used. The variable involved in the second goal is the first instance that satisfies the corresponding secondary effect of the operator. SIPE-2 solves the sequential problem but it does not solve the parallel problem.

SIPE-2 fails to solve 10 (out of 13) of the parallel problems displayed in the table shown below. The three parallel problems that SIPE-2 could solve had the following characteristics: (1) either the instantiate slot or the copy operator was used; (2) the order of the instance involved in the goal was 1st. SIPE-2 fails to solve all the parallel problems, even when using the instantiate slot or the copy operator, when the order of instance involved in the goal was 2nd. A possible explanation for SIPE-2 not to phantomize the goals has to do with the heuristics used in SIPE-2, preventing all the possible shuffles of parallel actions. SIPE-2 does not phantomize a goal when that goal was achieved by some operator that was used to achieve another parallel goal, at the same planning level. SIPE-2 is provided with the *instantiate* slot and the operator *copy* to overcome situations like this. However, if the goal of the second parallel branch involves an instance that has 2nd order, i.e., it is the second instance in the file that satisfies the goal, a different problem occurs. An explanation for this might be the fact that SIPE-2 instantiates the variable in the goal to be phantomized but does not phantomize variables in the plan to make a phantom. The problem seems to be that the TC (Truth Criterion) collects constraints to put on the query variables, rather than on variables in the plan.

[4]The process of the operator *copy* is the action *copy* which copies the same goal to the next planning level instead of expanding it. Using the operator copy might introduce some problems. One has to worry about having goals phantomized at the same level as other parallel goals, which sometimes is not obvious. Furthermore, the operator copy might result in a loop, rather than phantomizing the goal.

GOAL PHANTOMIZATION ANALYSIS IN SIPE-2

# Instances (satisfying constraints)	Order of Instance	Instantiate Slot	Copy Operator	Sequential/ Parallel
-	N/A	NO	NO	Sequential
-	N/A	NO	YES	Both
1	1st	NO	NO	Sequential
1	1st	YES	NO	Sequential
1	1st	NO	YES	Both
1	1st	YES	YES	Both
2	1st	NO	NO	Neither
2	1st	YES	NO	Sequential
2	1st	NO	YES	Neither
2	1st	YES	YES	Both
2	2nd	NO	NO	Neither
2	2nd	YES	NO	Neither
2	2nd	NO	YES	Neither

```
;;; definition of the objects in the world

(DEFINE.DOMAIN)
CLASS: PERSONS
INSTANCES:  Nelson,  Carla, Ken;
END CLASS


CLASS: PLACES
INSTANCES:  ROME, NY
END CLASS

STOP

(DEFINE.DOMAIN)
PREDICATES:

(nice Nelson)
(on Nelson NY)
(on Carla NY)
(on Ken NY)

END PREDICATES
STOP
(DEFINE.DOMAIN)

OPERATOR: transport-person-ride;
ARGUMENTS: person1,  place1, person2 is not person1, place2 is not place1;
PURPOSE: (on person1 place1) ;
PRECONDITION: (nice person1), (on person1 place2), (on person2 place2);
INSTANTIATE: person2, place2;
PLOT:

    PROCESS
ACTION: transport-person-prim;
        ARGUMENTS: person1, person2,  place2, place1;
        EFFECTS: (on person1 place1), (not (on person1 place2)),
                 (on person2 place1), (not (on person2 place2));


END PLOT END OPERATOR
```

```
STOP

(DEFINE.PROBLEM)

PROBLEM: prob-ride

GOAL: (on Nelson Rome)
        Goal: (on Carla Rome)



END PROBLEM
PROBLEM: prob-ride-parallel
Parallel

Branch 1: GOAL: (on Nelson Rome)
        Branch 2: Goal: (on Carla Rome)
End Parallel

END PROBLEM
STOP
```

O-Plan2 solves all of the 13 problems, either when the goals are in parallel or sequential. In order to satisfy a goal (an *achieve* condition in O-Plan2 language), the QA (Truth Criterion) uses the tactics expand an operator, link-with-binding, link-no-binding, already-satisfied and always statements. In this particular case, O-Plan2 uses the tactic link-with-binding in order to satisfy the condition, even if the goal was achieved by some operator that was used to achieve another parallel goal, at the same planning level. A major concern of the O-Plan2 team has been to design a system able to find any solution to condition achievement problems and which has a systematic procedure for doing this in its QA rather than relying on heuristic tactics.

7 Numerical Reasoning

One of the main reasons why we selected the MC puzzle to be encoded in SIPE-2 and O-Plan2 was that it deals with (simple) numerical calculations. We were particular curious to test the way the planners deal with numerical variables, especially having in mind that they emphasize a least commitment approach.

The current version of O-Plan2 does not perform numerical calculation.

Among the AI planners, SIPE-2 is one of the pioneers in manipulating numerical quantities. SIPE-2 considers two classes of numerical variables, *numerical* and *continuous*. The numerical class is used for variables that eventually instantiate to one particular value. The continuous class is used for variables that will have values changing over time. SIPE-2 provides the special predicates *level*, *produce*, and *consume* for numerical reasoning. Furthermore, SIPE-2 allows the usage of function constraints on variables. In order to alleviate the complications introduced by function constraints on numerical variables, SIPE-2 is provided with heuristics. When uninstantiated numerical variables have function constraints, SIPE-2 computes a range for the value of the variable by calling the function on all the possible instantiations that are currently consistent. However, this approach is not efficient in large domains. Wilkins [Wilkins 88] suggests the substitution of the range by an estimate, without developing that idea. I do not know how this approach would be implemented when encoding a problem in SIPE-2. SIPE-2 also uses summary-range constraints to avoid recomputation of numerical quantities. These constraints summarize in one numerical range all the consequences of the other constraints on a numerical variable. Basically, the idea is to store the results of computing a numerical variable's constraints by placing a summary-range constraint on the variable. However this procedure cannot be done for continuous variables since their values vary with time.

Discussions with other researchers raised the issue that the MC puzzle might be a "bad" example for partial order planners which rely heavily on a least commitment approach, due to its constraints defining a dependency between all the variables of the problem. However, constraints of the type of the MC puzzle are very common in real world problems, for instance in the transportation domain. Typical AI approaches to dealing with constraints involving several variables are Constraint Satisfaction Problems (CSP). It is interesting to notice that CAMPS, a planner more tailored to reasoning about resources, formulates its problems as CSP. An issue related to this is the role of generative planners and the integration of planning and scheduling.

My experience with numerical reasoning in SIPE-2 is small, limited to the MC puzzle and other small problems. Nevertheless, considering the problems that we had to implement the numerical constraints in SIPE-2[5], considering that the current version of O-Plan2 does not

[5]The implementation of the MC puzzle revealed differences in Lucid Lisp numerical routines that had not been previously exercised in the ported Sun version. David Wilkins says that most of the problems that we encountered did not exist in the Symbolics version.

handle numerical constraints, I feel that SIPE-2 and O-Plan2 should be more extensively tested in numerical domains.

8 Search

Search in O-Plan2 and SIPE-2 is a search through a space of partial plans rather than a search through a space of world st. s. The search control strategy embodied in the current version of O-Plan2 and SIPE-2 is very simple. Both systems implement a depth-first search, with chronological backtracking.

Both systems consider the possibility of encoding search-control mechanisms within the operators, through the use of preconditions or condition types or by implementing some type of metalevel operators. However, there is no defined domain-independent methodology or approach to embody search-control mechanisms within the operators or to design metalevel operators. For the O-Plan2 team the issue of encoding search-control mechanisms through the use of condition types is still an open question. I do not see much difference between the current use of condition types in O-Plan2 and the use of preconditions in SIPE-2 as a form of guiding search.

O-Plan2 guarantees to produce at least one valid solution to a given problem if that is feasible within the constraints specified on the task and within the modeling capabilities provided by the constraint managers installed. However, O-Plan2 does not guarantee to produce more than one such valid solution for any problem. O-Plan2 is not suitable for problems in which all (syntactically different) solutions are required or in which an optimal solution is needed. This approach to defining the search space of a planner was first introduced in INTERPLAN and subsequently used in Nonlin and O-Plan2. On the other hand, SIPE-2 uses heuristics to guide search which means that it does not guarantee that a solution will be found, if it exists. I presented several cases in section 6 whereby SIPE-2 fails to solve feasible problems. It would be important to investigate how the heuristics embodied in SIPE-2 perform in general, as well as the trade off between the heuristic approach embodied in SIPE-2 and the complete search embodied in O-Plan2, in particular regarding time performance and number of cases where SIPE-2 fails to find a solution when that solution was found by O-Plan2.

Constraint propagation is another mechanism intended to be provided in O-Plan2 to alleviate search [Tate *et al* 94a, Drabble & Tate 94]. Basically, the idea is whenever decisions are performed at certain nodes concerning the values of variables, the domain of other variables that are involved in common constraints is redefined by the propagation of effects. Examples of constraint managers in O-Plan2 include: Resource Utilization Manager (RUM) and Time Point Network Manager (TPNM). *Constraint Managers* have as main role to maintain information about a plan while it is being generated. The information can then be used to prune search (where plans are found to be invalid as a result of propagation of the constraints managed by the these managers) or to order search alternatives according to some heuristic priority. Constraint Managers are intended to provide efficient support to a higher level of the planner decision where decisions are taken. They should not take any decision themselves. They are intended to provide complete information about constraints they are managing. The current version of O-Plan2 only exploits constraint propagation partially, in terms of propagation of time windows. However, we have not tested this feature. Regarding resource

reasoning, the current version of O-Plan2 only has a very simple model of resource reasoning, a model for strict consumable resources. However, one of the features that O-Plan2 claims to be innovative about is its rich model for resource reasoning, to be implemented in future versions of O-Plan2. The concepts of resource reasoning to be implemented in O-Plan2 in its Resource Utilization Manager are inspired by the models used by schedulers such as OPIS [Smith *et al* 90] and TOSCA [Beck 93], a blackboard type framework developed at AIAI, Edinburgh, for job shop scheduling. Information about resource utilization is used to prune search. RUM also provides a mechanism for the incremental management of optimistic and pessimistic resource usage profiles in an activity planning framework. I consider that it is important to experiment the future releases of O-Plan2 to analyze how these concepts perform in particular to analyze the trade off between the quality of the solutions and the time performance of the system.

In SIPE-2, there is no concept of constraint propagation to guide the search. Plan critics (e.g., for resource reasoning and temporal reasoning) have the main function of checking the feasibility of the solution rather than guiding the search. In SIPE-2, temporal reasoning and resource reasoning are not used to guide the search.

Interaction with the users throughout the planning and plan execution processes is another mechanism to guide search. This mechanism is intended to allow users to address larger problems that may initially be beyond the capabilities of fully automatic planning techniques and it is also useful for debugging. The user interface of the current version of O-plan2[6] gives emphasis to the way the different modules of the blackboard interact. This feature is useful for debugging process but it is not very useful for a user that is not a programmer. The mechanisms to facilitate interaction with the user are better in SIPE-2, and they allow an easier understanding of what is going on for a non-programmer user. Nevertheless, even though SIPE-2 is provided with a better user interface, neither of the planners address that issue in depth.

[6] I am referring to the version of O-Plan2 currently installed on Rome Laboratory machines, without the AUTOCAD interface. I have not seen O-Plan2 with the AUTOCAD interface.

9 Conclusions

This paper reports some experiments that were performed at Rome Laboratory using two generative planners: O-Plan2 ([O-Plan2 93], [Currie & Tate 91]) and SIPE-2 ([Wilkins 93], [Wilkins 88]). The team that participated in the project consisted of: Carla O.Ludlow, as consultant to Rome Laboratory, Karen Alguire and Albert Franz, from Rome Laboratory. Albert Franz was involved in the project for only two months. The main objective of the project was to provide the in-house team with an understanding of generative planners and familiarization with O-Plan2 and SIPE-2. A comparison between O-Plan2 and SIPE-2 was a secondary objective, mainly a subproduct of the experimentation with both planners.

In an attempt to summarize my conclusions in one paragraph I would say that SIPE-2 is definitely more mature than O-Plan2: the current version of SIPE-2 is version 4.2 while the current version of O-Plan2 is version 2.1. O-Plan2 is still an ongoing project. That is reflected in the fact that even basilar concepts of O-Plan2 are still under discussion. For example, in September 93 when I was trying to fully understand the differences between each condition type, I was told by the O-Plan2 team that condition types are still an area of research (and even change) in O-Plan2. The difference of maturity between the systems is reflected on the functionality that each systems offers. O-Plan2's functionality is a subset of the functionality offered in SIPE-2, which can be seen in the fact that O-Plan2 does not perform numerical reasoning, O-Plan2 does not include deduction of context dependent effects, O-Plan2 only deals with a very simple model of strictly consumable resources, etc. Version 2.2 of O-Plan2 with resource oriented demonstration is planned to be delivered in July 94 and version 2.3, which integrates planning, execution and re-planning, is planned to be delivered in July 95. Nevertheless, SIPE-2 is also a research product with room for improvement. It needs further testing. Just as a result of our encoding exercise of the MC puzzle in SIPE-2 several errors were detected and several patches were added to the system. Furthermore, there are several limitations or weaknesses in SIPE-2 as pointed out in the report.

One aspect that we examined as part of our experiments was deductive rules. A powerful feature of SIPE-2 is the possibility of encoding a causal theory to represent and reason about the effects of actions in different world states and therefore it allows separation of knowledge about actions from knowledge about causality. O-Plan2 has no implemented mechanism to represent a theory of causality and it is not planned to add that feature to the future releases of O-Plan2. This is a strong limitation, with repercussions in terms of search and time necessary to encode problems. As a result of the lack of a mechanism to automatically deduce context dependent effects, all the effects of an action have to be explicitly stated. Furthermore, actions that have different effects depending on the particular state of the world have to be encoded through a multiplicity of operators, each operator corresponding to a possible situation in which the action might take place.

The current version of O-Plan2 does not perform numerical calculation. Among the AI planners, SIPE-2 is one of the pioneers in manipulating numerical quantities. Nevertheless,

it is not clear to me how SIPE-2 scales up. Wilkins [Wilkins 88] suggests the usage of estimates when dealing with larger problems, without developing that idea. This aspect requires further testing.

Another aspect that we looked at was goal phantomization. I presented several situations in section 6 whereby SIPE-2 fails to phantomize goals as a result of its heuristics. O-Plan2 is able to find solutions for all of the problems that SIPE-2 could not solve.

O-Plan2 guarantees to produce at least one valid solution to a given problem if this is feasible within the constraints specified on the task and within the modeling capabilities provided by the constraint managers installed. In order to alleviate search, O-Plan2 team envisages the implementation of constraint propagation mechanisms, in particular for temporal reasoning[7] and resource reasoning. However, O-Plan2 does not guarantee to produce more than one such valid solution for any problem. O-Plan2 is not suitable for problems in which all (syntactically different) solutions are required or in which an optimal solution is needed. In SIPE-2 there is no concept of constraint propagation to guide the search. Plan critics (e.g., for resource reasoning and temporal reasoning) have as main function to check the feasibility of the solution rather than to guide the search. In SIPE-2 temporal reasoning and resource reasoning are not used to guide the search.

Interaction with the users throughout the planning and plan execution processes is another mechanism to guide search. This mechanism is intended to allow users to address larger problems that may initially be beyond the capabilities of fully automatic planning techniques and it is also useful for debugging. The user interface of the current version of O-Plan2[8] gives emphasis to the way the different modules of the blackboard interact. This feature is useful for debugging process but it is not very useful for a user that is not a programmer. The mechanisms to facilitate interaction with the user are better in SIPE-2, and they allow an easier understanding of what is going on for a non-programmer user. Nevertheless, even though SIPE-2 is provided with a better user interface, neither of the planners address that issue in depth.

Regarding search mechanisms, it would be important to investigate how the heuristics embodied in SIPE-2 perform in general, as well as the trade off between the heuristic approach embodied in SIPE-2 and the fact that O-Plan2 guarantees to find a solution if it exists, in particular regarding time performance and number of cases where SIPE-2 fail to find a solution when that solution was found by O-Plan2.

As a final remark, I would like to stress the importance of the search mechanisms when dealing with NP-complete problems and NP-hard problems as it is the case of most planning problems. Several strategies can be adopted, e.g., user interaction, constraint propagation, intelligent backtracking, reason maintenance, variable ordering for instantiation, order for the instantiation of different values of a given variable. The question is what strategies

[7]The current version of O-Plan2 alreday has implemented mechanisms for constraint propagation of time windows. We have not tested it.

[8]I am referring to the version of O-Plan2 currently installed on Rome Laboratory machines, without the AUTOCAD interface. I have not seen O-Plan2 with the AUTOCAD interface.

to combine and what is the best mix of search strategies. This issue is a central topic of investigation. Intelligent search control for generative planners is an area that requires further research.

10 Acknowledgements

I thank Nort Fowler for the helpful discussions throughout the project. I also thank Austin Tate, David Wilkins, Brian Drabble and Jeff Dalton for their comments and cooperation throughout the project. Last, but not least, I thank Karen Alguire for her important contribution to the project. All views expressed in this paper are those of the author only.

References

[Alguire 94] Karen M. Alguire. Encoding user-defined problems with SIPE-2. To appear as a technical report of Rome Laboratory, 1994.

[Beck 93] Howard Beck. A Novel Approach to the Management of Job-shop Scheduling Constraints. In C. Kooij, P. A. MacConaill, and J. Bastos, editors, *Proceedings of the Ninth CIM-Europe Annual Conference on Expert Systems and the Leading Edge in Production Planning and Control*, pages 138–149, 1993.

[Currie & Tate 91] K. W. Currie and Austin Tate. O-Plan: the Open Planning Architecture. *Artificial Intelligence*, 51(1), Autumn 1991.

[Drabble & Tate 94] Brian Drabble and Austin Tate. The Use of Optimistic and Pessimistic Resource Profiles to Inform Serach in an Activity Based Planner. Rome Laboratory Progress Report,ARPA-RL/O-Plan2/TP/5 Version 1, 1994.

[Drabble 93] Brian Drabble. Conversion of SIPE-2 Domain Descriptions to O-Plan2 TF, September 1993.

[Ludlow & Alguire 94] Carla O. Ludlow and Karen M. Alguire. Looking at O-Plan2 and SIPE-2 Through Missionaries and Cannibals. To appear in the Proceedings of the ARPA-RL Planning Initiative Workshop, February 1994.

[O-Plan2 93] O-Plan2. *O-Plan2: the Open Planning Architecture, User Guide Version 2.1.* AIAI, University of Edinburgh, 1993.

[Smith *et al* 90] Stephen F. Smith, Peng Si Ow, Jean-Yves Potvin, Nicola Muscettola, and Dirk C. Matthys. An Integrated Framework for Generating and Revising Factory Schedules. *J. Opl Res. Soc.*, 41(6):539–552, 1990.

[Tate 93] Austin Tate. The Use of Condition Types in O-Plan2, September 1993.

[Tate *et al* 94a] Austin Tate, Brian Drabble, and Jeff Dalton. Reasoning with Constraints within O-Plan2. Rome Laboratory Progress Report - ARPA-RL/O-Plan2/TP/6 Version 1, 1994.

[Tate *et al* 94b] Austin Tate, Brian Drabble, and R.B. Kirby. O-Plan2: An Open Architecture for Command, Planning and Control. In M. Fox and M. Zweben, editors, *Knowledge-Based Scheduling.* Morgan Kaufmann, 1994.

[Wilkins & Desimone 92] David E. Wilkins and Roberto Desimone. Applying an AI Planner to Military Operations Planning. In M. Zweben and M. Fox, editors, *Intelligent Scheduling.* Morgan Kaufmann, 1992.

[Wilkins 88] David E. Wilkins. *Practical Planning: Extending the Classical AI Planning Paradigm.* Morgan Kaufmann Publishers, INc., 1988.

[Wilkins 93] David E. Wilkins. *Using the SIPE-2™ Planning System - A Manual for SIPE-2 Version 4.3.* SRI International, 1993.

*U.S. GOVERNMENT PRINTING OFFICE: 1994-510-117-50010

MISSION

OF

ROME LABORATORY

Mission. The mission of Rome Laboratory is to advance the science and technologies of command, control, communications and intelligence and to transition them into systems to meet customer needs. To achieve this, Rome Lab:

a. Conducts vigorous research, development and test programs in all applicable technologies;

b. Transitions technology to current and future systems to improve operational capability, readiness, and supportability;

c. Provides a full range of technical support to Air Force Materiel Command product centers and other Air Force organizations;

d. Promotes transfer of technology to the private sector;

e. Maintains leading edge technological expertise in the areas of surveillance, communications, command and control, intelligence, reliability science, electro-magnetic technology, photonics, signal processing, and computational science.

The thrust areas of technical competence include: Surveillance, Communications, Command and Control, Intelligence, Signal Processing, Computer Science and Technology, Electromagnetic Technology, Photonics and Reliability Sciences.